April 19, 2017

Automatically Managed Pools of Large Test Environments

Andrew Carter

Automatically Managed Pools of Large Test Environments

ABSTRACT

Large production-like test environments (sandboxes) are widely used to perform manual and automated code testing, feature validation, and evaluation of uncommitted code. Sandboxes represent the production jobs, datastores, and configurations needed to effectively replicate a production environment. However, sandboxes can be difficult to start, have a steep learning curve for configuration, and suffer from reliability concerns. Also, sandboxes can be challenging to share beyond an original engineering team, e.g., for integration testing. Due to such challenges, engineering time is wasted deploying sandboxes, and some development teams instead utilize live production environments for testing putting a product or service at risk.

This disclosure includes techniques to automatically provision pools of sandboxes for immediate use and ensure sandbox health and freshness. Sandbox pools described herein support automatic starting of new sandboxes, automatic clean-up of sandboxes returned by a user, and configurable lease lengths, after which a sandbox is automatically reclaimed. Configurable access control is provided at a pool level to control usage and enable sharing.

KEYWORDS:

- Sandbox
- Test environment
- Software testing
- Integration testing

BACKGROUND

Software engineering teams use large production-like test environments (sandboxes) to validate new product features, perform manual and automated code testing, and work with uncommitted code. Sandbox configurations are often specific to an engineering team. A sandbox represents a set of production jobs, datastores, and configurations that effectively replicate a live production environment that can be used to test applications prior to the applications being launched in live environments.

Sandboxes as used currently are difficult to start and have a steep learning curve for proper configuration. For example, knowledge of the complete software stack is necessary to successfully configure and start a sandbox. At times, sandboxes are unreliable, e.g., due to size and complexity. Also, in many situations, sandboxes are difficult to share across teams, e.g., for integration testing.

Due to such challenges, sandboxes are a source of wasted engineering time and frustration. Also, engineering teams may forego using a sandbox and instead, opt to test code in live production environments due to such difficulties. However, such testing imposes risks on the product or service, e.g., increased stability concerns due to the introduction of untested code.

DESCRIPTION

This disclosure describes techniques that implement managed sandbox pools. The techniques address difficulties associated with creation, configuration, maintenance, and sharing of sandboxes. The techniques automatically provision pools of sandboxes that enable engineering teams to check out individual sandboxes that can be put to immediate use without explicit configuration.

The pools of automatically managed sandboxes are designed to support several features. For example, a pool has a definable configuration that includes a size of the pool, e.g., total number of sandboxes, maximum number of idle sandboxes, etc. Further, the pool configuration specifies ownership and access control for the pool. Further, the pools are automatically managed e.g., to ensure that spare sandboxes are started as needed, refresh existing sandboxes and keep them healthy, track resource usage, provide notifications about available pool capacity etc. The managed pools also support command-line and/or web user interfaces that enable users to check out and return sandboxes from the pools.

The automatically managed pools support an arbitrary number of sandboxes. Each pool represents a different sandbox configuration and associated parameters. For example, a first pool (Pool 1) can be configured to use sandbox configuration A and support up to 10 sandboxes, with 3 sandboxes available for immediate checkout. A second pool (Pool 2) can be configured to use sandbox configuration B and support up to 50 sandboxes, with 25 sandboxes available for immediate checkout.

Pool configuration includes parameters that specify a time duration for which a sandbox from the pool may be checked out. For example, the configuration for pool 1 specifies that sandboxes may be checked out for up to 1 day. Further, pool 1 guarantees that any available sandbox is no more than 12 hours old. Similarly, the configuration for pool 2 specifies that sandboxes can be checked out for up to 2 weeks and that any available sandbox is no more than 36 hours old.

The pools of sandboxes are managed to ensure that sandboxes are running, healthy, and kept fresh. Further, the techniques include automatically starting new sandboxes to meet demand as sandboxes are checked out, up to a configurable maximum number of sandboxes that denotes a capacity of the pool. Also, the techniques provide configurable access control at the pool level to control check out of sandboxes and permit specific users to monitor resource allocation and usage.

Sandboxes within a pool are automatically managed. For example, sandboxes that are returned to the pool are automatically cleaned up, e.g., by stopping jobs, deleting datastores, etc. Also, when a lease period for a sandbox expires, the sandbox is automatically reclaimed. The techniques permit users to extend the length of a sandbox lease dynamically, if required.

By providing configurable administrative access control to sandboxes at the pool level, the techniques allow centralized monitoring to track resources and send notifications as resources are consumed or when resource constraints prevent creation of additional sandboxes. The sandbox pool configurations are highly customizable, e.g., parameters such as pool size, access control, lease length and freshness, etc. can be adjusted per need. Such customization enables use of the automatically managed pool of sandboxes for a variety of test environment needs.

The automatically managed sandbox pools per the techniques described offer several advantages. Sandboxes are available immediately upon checkout and users do not need to wait for sandboxes to start and become healthy. Further, the techniques eliminate the need for users to troubleshoot or debug sandbox startup failures. The learning curve to configure and start sandboxes is reduced, enabling users to grab a sandbox whenever needed. The techniques enable users without knowledge of the stack represented by the sandbox to deploy and use sandboxes for testing without significant training. The techniques also enable teams to centrally manage sandbox configurations and access.

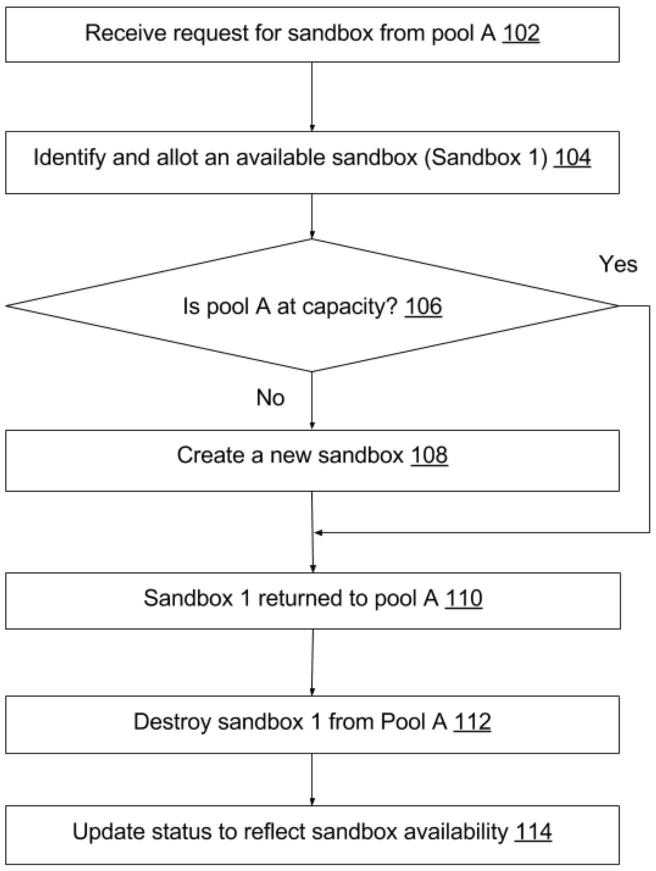
Receive request for sandbox from pool A 102
Identify and allot an available sandbox (Sandbox 1) 104
Is pool A at capacity? 106
Yes
No
Create a new sandbox 108
Sandbox 1 returned to pool A 110
Destroy sandbox 1 from Pool A 112
Update status to reflect sandbox availability 114

Fig. 1 Management of a sandbox pool

Fig. 1 shows an example process to manage a sandbox pool. A user requests checkout of a sandbox (102) from pool A, e.g., via a command-line interface or a web interface. In response, sandbox 1 is provided to the user (104). After providing a sandbox from pool A, it is determined whether pool A is at capacity (106). If pool A is not at capacity, a new sandbox is started (108) to replace the sandbox that was checked out. Next, the user returns sandbox 1 to pool A (110). Upon return, the returned sandbox is destroyed (112), as described above. The status of pool A is updated to reflect that the reclaimed sandbox is ready to be started and assigned to meet new demand (114).

User requests for sandboxes are received via a command-line or web interface. Further, commands to request and return sandboxes can be included in a script to check out multiple sandboxes, perform automated testing, and return sandboxes when testing is completed. Further, the user interface also displays a status of pools available to a user, e.g., a count of available sandboxes, freshness of sandboxes, sandbox configuration, number of active/ idle/ spare sandboxes within a pool, etc.

CONCLUSION

This disclosure describes automatically managed pools of sandboxes that address many challenges in deploying test environments. The techniques described herein enable sandboxes of a required configuration to be started quickly, automatically destroy sandboxes no longer in use, provide configurable lease times for sandboxes, and ensure sandbox freshness and health. User interfaces are provided to enable users to checkout and return sandboxes, to view sandbox availability, freshness, and health, and to view sandbox configurations available in various pools. Access control is implemented such that users can check out sandboxes of specific configurations, and pools are managed centrally.